Travaux pratiques 3

Modélisation et simulation d'un système de contrôle

IESE5/MISTRE – Polytech Grenoble

Objectif

Ce TP (et les suivants) sont consacrés à la conception d'un système de contrôle simplifié pour un four à micro-ondes.

Cette semaine, il vous est demandé **d'étudier très attentivement le système décrit ci-dessous, puis de compléter l'automate correspondant, de réaliser sa description VHDL et des simulations pertinentes**.

Le système est conçu de telle façon que l'utilisateur commande le four comme suit :

— Il choisit d'abord s'il veut le faire fonctionner le four à pleine puissance (600 Watts) ou moyenne puissance (300 Watts).

— Il choisit le temps de chauffage, puis commande le démarrage (pourvu que la porte du four soit fermée). Nous allons simplifier le choix du temps d'opération : il pourra seulement être de 30 secondes, 1 mn ou 2 mn.

— S'il commande l'ouverture de la porte pendant que le four est en fonctionnement, celui-ci s'arrête. Il est possible de le faire repartir après avoir refermé la porte.

Un compteur sera associé au système de contrôle : il démarrera lorsque l'utilisateur commandera le démarrage et comptera le temps écoulé (pour simplifier, nous pourrons supposer l'utilisation d'une horloge à 1 Hz, de telle sorte qu'un cycle d'horloge corresponde à 1 seconde). Si l'utilisateur commande l'ouverture de la porte pendant le chauffage, le compteur s'arrête, puis reprend lorsque la porte est refermée et la commande de démarrage renvoyée.

À la fin des séances de TP, il vous sera demandé de rendre (par mail) :

— L'ensemble de vos fichiers sources commentés (uniquement les `.vhd`)

— Un rapport du projet complet contenant une description détaillée de chaque code que vous avez réalisé, une description de chaque simulation, ainsi qu'une justification des résultats (positifs ou négatifs) que vous avez obtenu à chaque étape de chaque TP (TP3 et TP4).

Attention : demandez à votre enseignant la fiche des conseils pour la réalisation du rapport.

1 Le système complet

Le **système complet** est formé d'un contrôleur et d'un compteur, interconnectés de la façon suivante (voir détails ci-dessous) :

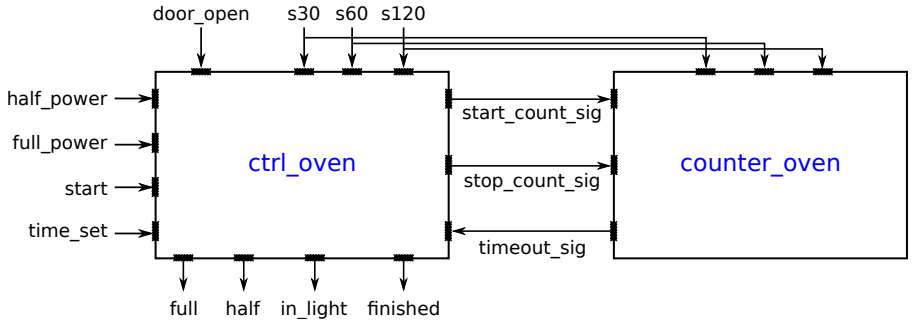

Figure 1 – Système complet (four à micro-ondes)

1.1 L'entité VHDL `ctrl_oven`

L'entité VHDL **`ctrl_oven`** ci-dessous décrira l'interface de ce système de contrôle :

```
entity ctrl_oven is
  port(
     clk, rst, half_power, full_power, start, s30, s60, s120,
     time_set, door_open, timeout : in std_logic;
     full, half, in_light, finished, start_count, stop_count : out std_logic
  );
end entity;
```

Il est synchronisé par les fronts montants de l'horloge `clk`, et est muni d'un reset asynchrone actif haut `reset` (non montrés sur le dessin, pour simplifier). Les autres entrées primaires jouent les rôles suivants : `half_power` correspond au bouton servant à demander une puissance de 300 Watts, `full_power` correspond au bouton servant à demander une puissance de 600 Watts (évidemment, seul 1 bouton au plus peut être enfoncé à tout moment), `start` correspond au bouton servant à commander le démarrage, `s30`, `s60` et `s120` sont aussi des sélecteurs qui ne peuvent pas être activés simultanément et qui contrôlent le temps de chauffage (`30s`, `1mn`, ou `2mn`), `time_set` correspond au bouton servant à indiquer que le temps a été choisi, `door_open` est une entrée en provenance d'un capteur qui indique si la porte est ouverte, et `timeout` est l'entrée en provenance du compteur indiquant que la fin de chauffage est atteinte.

Les sorties primaires `full` et `half` permettent de commander l'allumage de leds pour indiquer la puissance choisie, `in_light` commande l'allumage de la lumière à l'intérieur du four, `finished` commande l'allumage d'une led pour indiquer que la cuisson est finie, `start_count` et `stop_count` sont des interfaces avec le compteur (voir plus loin).

L'automate possède un état initial `idle` (dans lequel le système revient en cas de reset asynchrone), qui conduit à l'état `full_power_on` si `full_power` est sélectionné (entrée à `'1'`) et à l'état `half_power_on` si `half_power` est sélectionné. Une fois dans l'un de ces états, l'utilisateur peut encore changer son choix en changeant sa sélection (`half_power` ou `full_power`), l'automate s'adaptant en conséquence. S'il ne change plus de bouton, mais active l'un des sélecteurs `s30`, `s60` ou `s120`, l'automate passe dans un état `set_time` dont il ne sortira que lorsque l'utilisateur appuyera sur le bouton `time_set` pour confirmer qu'il a réalisé son choix de temps. A ce point, si la porte du four est fermée, l'automate transite vers un état `operation_enabled`, et sinon il atteint l'état `operation_disabled`. Depuis l'état `operation_enabled`, l'automate rejoint l'état `operating` si l'utilisateur commande le démarrage. Depuis l'état `operation_disabled`, l'automate ne pourra que rejoindre l'état `operation_enabled` lorsque l'utilisateur refermera la porte du four.

L'état `operating` conduira à l'état `complete` lorsque le temps voulu sera écoulé (fin de chauffage atteinte), mais pourra ramener à l'état `operation_disabled` si l'utilisateur ouvre la porte. Dans ce cas, il faudra alors repasser par l'état `operation_enabled` pour pouvoir rejoindre l'état `operating` et continuer l'opération. Une fois l'état `complete` atteint, l'ouverture de la porte remettra le système dans son état initial.

La sortie `full` passe à `'1'` pour indiquer que la pleine puissance a été choisie (c'est à dire sur les transitions conduisant à `full_power_on`) afin d'allumer la led correspondante. De même, la sortie `half` passe à `'1'` pour indiquer que la demi-puissance a été choisie (c'est à dire sur les transitions conduisant à `half_power_on`) afin d'allumer la led correspondante.

La sortie `finished` passe à `'1'` pour indiquer que l'opération est finie (c'est à dire sur les transitions conduisant à `complete`) afin d'allumer la led correspondante. La lumière intérieure du four est allumée (sortie `in_light` à `'1'`) lorsque la porte du four est ouverte (c'est à dire sur les transitions conduisant à l'état `operation_disabled`) ou lorsque le système est en opération (c'est à dire sur les transitions conduisant à `operating`).

La sortie `start_count` sert à déclencher le compteur, pour ce faire elle passe à `'1'` dans le cycle conduisant de `operation_enabled` à `operating`. De même la sortie `stop_count` sert à arrêter le compteur si la porte s'ouvre, pour ce faire elle passe à `'1'` dans le cycle conduisant de `operating` à `operation_disabled`.

Question 1.1. Complétez le dessin de l'automate fourni ci-dessous (à inclure dans votre rapport).

Attention : bien faire apparaître sur le dessin le déterminisme de l'automate, selon les priorités énoncées ci-dessus, et les cas de rebouclage sur les états. Faire également apparaître les affectations des sorties. On rappelle que, par contre, les informations relatives au reset asynchrone n'apparaitront pas sur ce dessin, mais devront être prises en compte dans la description VHDL.

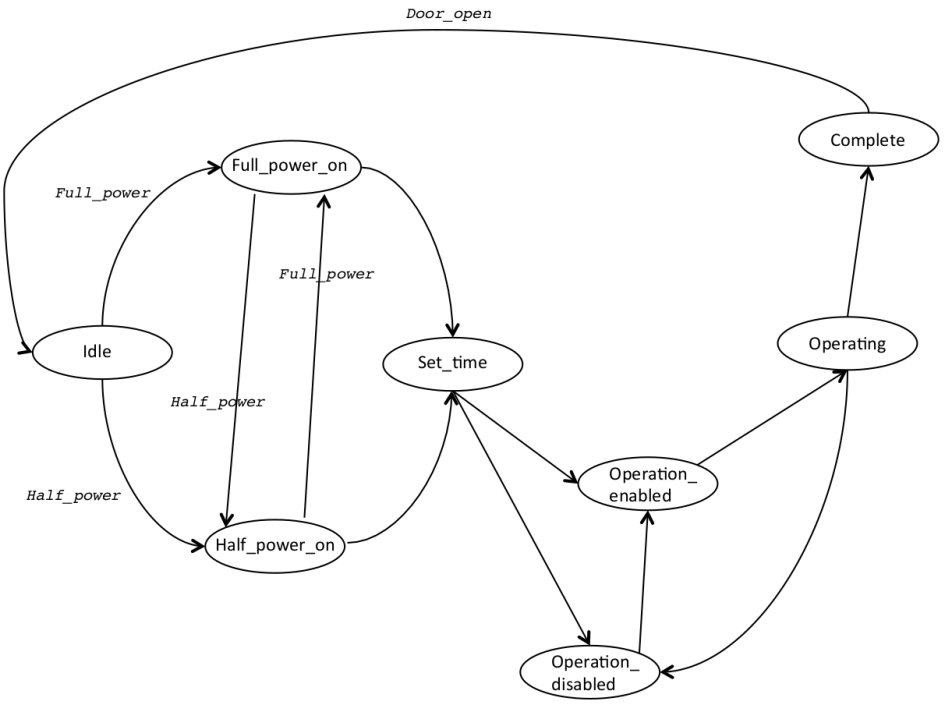

Figure 2 – Automate à compléter (four à micro-ondes)

Question 1.2. Écrivez soigneusement la description VHDL comportementale de ce système ("processes" de calcul de l'état suivant, de mise à jour de l'état, et de mise à jour des sorties).

Question 1.3. Concevez une entité de test illustrant divers scénarios (à expliquer en détail), en supposant une fréquence d'horloge de 50 MHz. Réalisez la simulation avec ModelSim et expliquez les résultats.

Attention : ne pas oublier de placer le circuit dans son état initial en provoquant un reset en début de simulation.

1.2 L'entité VHDL `counter_oven`

L'entité VHDL **`counter_oven`** ci-dessous décrira l'interface du compteur :

```
entity counter_oven is
  port(
     clk, rst, start, stop, s30, s60, s120 : in std_logic;
     aboveth : out std_logic
  );
end entity;
```

Les entrées `start` et `stop` proviennent du contrôleur (voir fonctionnement ci-dessus) et la sortie `aboveth` sera connectée au contrôleur pour lui indiquer que la fin de chauffage est atteinte.

Ce composant devra permettre de compter le temps à partir du moment où il reçoit `'1'` sur `start`. En supposant une fréquence adéquate, on pourra simplement compter les tops d'horloge et positionner la sortie `aboveth` à `'1'` lorsque la valeur voulue (`30` ou `60` ou `120`) sera atteinte. Ce compteur aura un registre interne `c` de type natural servant à compter, et 3 états symboliques : `idle` dans lequel `c` est maintenu à `0`, `counting` dans lequel `c` est incrémenté, mais dont on sort, si `stop` est reçu, pour aller dans un état `stopped` dans lequel le compteur est stoppé en attendant de recevoir `start` à nouveau et de revenir dans `counting`. Une description VHDL incomplète pour ce compteur vous est fournie à continuation :

```
library IEEE;
use IEEE.std_logic_1164.all;

entity counter_oven is
  port(
    clk, rst, start, stop, s30, s60, s120: in std_logic;
    aboveth: out std_logic
  );
end entity;

architecture impl of counter_oven is

  type States is (idle, counting, stopped);

  signal state, nextstate: States;
  signal c, nextc: natural;

  begin

    process (state, c, start, stop, s30, s60, s120)
    begin

      case state is
       when idle =>
         if start='1' then nextstate <= counting;
          nextc <= -- fill here c+1
         else nextstate <= idle;
          nextc <= -- fill here 0
         end if;
```

```
      when counting =>
        if stop = '1' then
          nextstate <= stopped;
          nextc <= -- fill here c
        elsif ((s30='1' and c >= 30) or (s60='1' and c >= 60) or
              (s120='1' and c >= 120)) then
          nextstate <= idle;
          nextc <= -- fill here 0
        else
          nextstate <= counting;
          nextc <= -- fill here c+1
        end if;

      when stopped =>
        if start='1' then
          nextstate <= counting;
          nextc <= -- fill here c+1
        else nextstate <= stopped;
          nextc <= -- fill here c
        end if;
    end case;
  end process;

  process (reset, clk)
  begin
    -- if asynchronous reset
    if (reset = '1') then
      state <= idle;
      c <= -- fill here 0
    -- if rising edge
    elsif (clk'event and clk='1') then
      state <= nextstate;
      c <= -- fill here nextc
    end if;
  end process;

  aboveth <= '1' when ((s30='1' and c >= 30) or (s60='1' and c >= 60) or
           (s120='1' and c >= 120)) else '0';

end architecture;
```

Question 1.4. Complétez la description VHDL fournie et, autant que possible, la tester séparément.

1.3 L'entité VHDL du système complet

La description du système complet sera composée par le contrôleur (`ctrl_oven`) et le compteur (`counter_oven`). L'architecture VHDL du système sera donc une architecture de type structurel (voir Figure 1).

Attention : Les entrées de l'horloge et du reset sont communes avec celles du contrôleur, de même que les entrées s30, s60 et s120.

Question 1.5. Réalisez la description VHDL du système formé par l'interconnection du contrôleur et du compteur, et concevez également une entité de test illustrant divers cas de figure (expliquez soigneusement dans cotre rapport les scénarios représentés).

2 L'analyse de couverture

Nous allons pouvoir évaluer la qualité des tests réalisés grâce à une analyse de couverture sur notre description d'automate (dans cette partie vous considérerez uniquement l'automate du contrôleur).

La description d'automate doit avoir été compilée avec des options préparant à l'analyse de couverture :

— Il faudra utiliser `+cover` suivi d'une ou plusieurs lettres, parmi elles `b` (branch) vise la couverture de branchement, `f` (fsm) la couverture de machine à états finis, et `s` (statement) la couverture d'instructions.

— Il faudra ensuite lancer `vsim` avec l'option `-coverage`.

Question 2.1. Simulez votre modèle et **générez le rapport de couverture** correspondant. Discutez en détail les résultats obtenus.

— Démarrez la simulation (`Simulate` → `Start Simulation`) comme d'habitude, et dans l'onglet `sim` à gauche double-cliquez sur le nom de l'instance associée à l'automate, ce qui aura pour effet d'ouvrir le fichier VHDL correspondant dans un nouvel onglet à droite. Les informations détaillées sur la couverture apparaitront plus tard dans cet onglet (Figure 3).

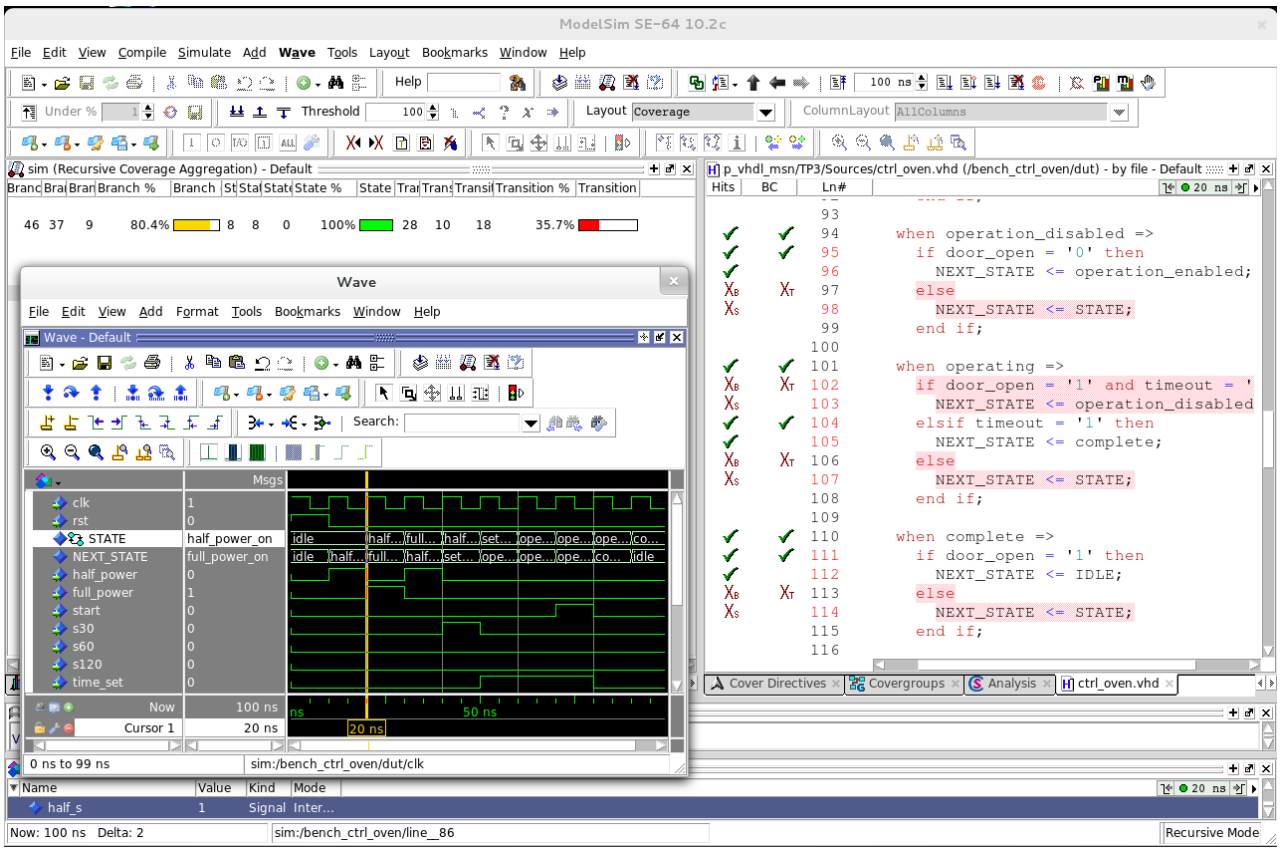

Figure 3 – Analyse de couverture

— Placez tous les signaux voulus dans la waveform et lancez la simulation. Hormis vos résultats habituels de simulation dans la waveform, vous obtiendrez :

  — dans l'onglet `sim` (à gauche), des informations globales sur les taux de couverture.

  — dans l'onglet contenant le source VHDL (à droite), des informations détaillées de couverture par instruction

Le symbole vert $\surd$ indique que la ligne a été couverte, le symbole rouge X indique le contraire. Dans la colonne `BC` (Branch Coverage), un X_T (ou X_F) indique que la branche true (ou false) d'une instruction conditionnelle n'a pas été exécutée. Si vous placez votre curseur sur une telle ligne, les symboles des colonnes `Hits` et `BC` deviennent des nombres, qui indiquent le nombre de fois où la ligne a été exécutée. Vous pouvez sauver un résumé de l'analyse de couverture avec `Tools` → `Coverage report` → `Text`.

Question 2.2. Utilisez le rapport de couverture obtenu pour améliorer (si nécessaire) vos jeux d'essai afin de valider le comportement de votre description. Décrivez chacune des transformations faites sur votre testbench et les améliorations obtenues.

Attention : Chaque amélioration proposée doit être soigneusement commentée dans votre rapport final, ainsi que les résultats obtenus (avant et après chaque changement) pour chaque type de couverture. Dans ce projet, on vous demande d'obtenir un taux de couverture très proche du 100%.

Notez également que vous pouvez utiliser le **parcours d'états** pour améliorer vos jeux d'essai :

— Trouvez la petite icône à côté de l'identifiant de votre état (Figure 4)

Figure 4 – Bouton `state`

— Cliquez sur l'icône afin d'ouvrir un nouvel onglet dans lequel des informations sur le parcours d'états seront données graphiquement en cours de simulation (Figure 5).

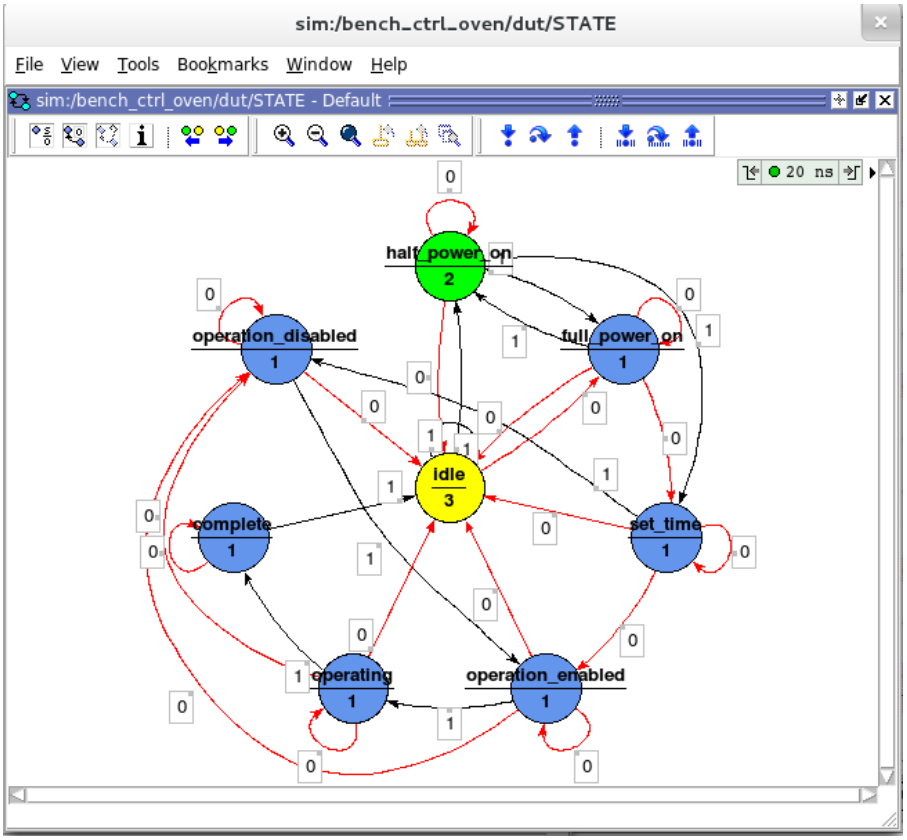

Figure 5 – Parcours d'états